White Paper

Nokia and Symbian OS

NOKIA
CONNECTING PEOPLE

Contents

Symbian is key to the future of the telecommunications industry. Nokia is basing its future smartphones on Symbian OS and it forms the basis of the recently launched Series 60 platform.

Nokia is a founding member and shareholder of the Symbian alliance.

This paper provides a basic understanding of why Nokia is strongly committed to Symbian – from both a commercial and a technical perspective. Additionally, some of the technical qualities of Symbian OS are examined in some detail – this section is aimed at the more technically minded reader.

Nokia and Symbian – the history – extracts from Nokia announcements

24.6.1998
"It was announced today that Ericsson, Nokia and Psion have conditionally agreed to form a new joint venture called Symbian. Nokia's investment is approx. USD 50 million. This agreement is further strengthened by the support of Motorola who have signed a Memorandum of Understanding to join Symbian."

21.11.2000
The Nokia 9210 Communicator launched – "Symbian's EPOC operating system bring open development interfaces to the Nokia 9210 Communicator for numerous additional applications to be provided by any third party developers," Anssi Vanjoki, Executive Vice President, Nokia Mobile Phones.

21.5.2001
"Nokia expects 50% of its 3G phones to use the Symbian Operating System by 2004." Jorma Ollila, CEO Nokia.

5.6.2001
The Nokia 9290 Communicator (for US) launched – "The 9290 Communicator demonstrates Nokia's commitment to not only provide unique, innovative products for our customers, but to do it using open standards, such as the Symbian OS, Java and SyncML" Paul Chellgren, Vice President of Business Development for Nokia.

12.11.2001
Nokia introduced the Series 60 Platform for application and feature driven mobile devices. The new platform is designed for Symbian OS and will support mobile browsing, multimedia messaging and content downloading, as well as a host of personal information management and telephony applications.

13.11.2001
Open Mobile Architecture alliance launched – AT&T Wireless, Cingular Wireless, MM02, NTT DoCoMo, Telefonica Moviles, Vodafone, Fujitsu, Matsushita, Mitsubishi Electric, Motorola, NEC, Nokia, Samsung, Sharp, Siemens, Sony Ericsson, Toshiba and Symbian to commit to products and services based on open mobile architecture enablers.

19.11.2001
The Nokia 7650 launched – "The Nokia 7650 delivers a full range of functions for professional needs in the EGSM900/1800 environment. Symbian OS allows the phone to be customized and upgraded by corporations and individual users, who will be able to purchase numerous add-on software applications designed by independent software developers."

What is Symbian

The Company
Headquartered in London, Symbian Ltd is owned by Ericsson, Nokia, Panasonic, Psion, Siemens and Sony-Ericsson.

Customers
Symbian's customers include all of its shareholders, but any company is free to license the product – Symbian OS is open to all on equal terms. So far, in addition to the shareholders, Sony, Sanyo, Kenwood and Fujitsu have all taken licenses.

Business model
The Symbian business model is simple – manufacturers pay a fee to Symbian for each device that they sell that uses Symbian OS. Symbian also earns money working with licensees to develop their products.

Basic principles
Nokia is committed to open platforms – in the area of Operating Systems as in many other areas. The cornerstone of Symbian's modus operandi is to use open – agreed – standards wherever possible. Symbian is focussed squarely on one part of the value chain – providing the base operating system for mobile internet devices. This enables manufacturers, networks and application developers to work together on a common platform.

Symbian history
Symbian OS started life as EPOC – the operating system used for many years in Psion handheld devices. When Symbian was formed in 1998, Psion contributed Epoc into the group. Epoc was renamed Symbian OS and has been progressively updated, incorporating both voice and data telephony technologies of ever greater sophistication with every product release.

Symbian OS

By setting the standard for wireless computing and telephony, Symbian brings together the wireless value chain. Symbian OS drives standards for the interoperation of data-enabled mobile phones with mobile networks, content applications and services:

A platform for wireless services
Symbian delivers an advanced, open, standard operating system to its licensees. Symbian OS is flexible and scalable enough to be used in the variety of mobile phones needed to meet a wide range of user requirements. Symbian OS supports complex requirements of network protocols worldwide and enables a broad, international developer community.

Providing wireless services
Open standards ensure global network interoperability, allowing mobile phone users to communicate with anyone, any way, at any time. The compelling advanced data services that operators can provide on Symbian OS phones will help minimize churn and maximize revenue.

Developing wireless services
Software developers are able, for the first time, to build applications and services for a global mass-market of advanced, open, programmable, mobile phones. A set of standard application programming interfaces (APIs) across all Symbian OS phones and the advanced computing and communications capabilities of Symbian OS, enable development of advanced services.

Symbian OS is a powerful aligning force for the wireless value chain. Mobile phone manufacturers, network operators and software developers are assured that they are working with an industry standard, open operating system that allows customization and is focused on the mass-market, driving the wireless community.

symbian

Symbian OS and Nokia products

Nokia 9200 Communicator Series
In June 2001, Nokia shipped the Nokia 9210 Communicator. This was the third Nokia Communicator, but the first based on Symbian OS. The full 9200 Communicator range now includes:

- Nokia 9210 Communicator
- Nokia 9210c Communicator – a chinese language version
- Nokia 9290 Communicator – designed for the Americas
- Nokia 9210i Communicator – an upgraded version of the original 9210

Nokia 7650
Nokia launched the Nokia 7650 in November 2001. This is the first Symbian OS phone to feature the "always on" capability of GPRS. With its built in camera, the Nokia 7650 is the first Nokia phone to feature Multimedia Messaging (MMS).

Series 60 Platform
In November 2001, Nokia launched the Series 60 Platform – which provides licensees with the ability to make advanced mobile phones. Licensees will be other phone manufacturers, who will be able to take advantage of Nokia technology to short-cut the usual development process.

Series 60 Platform builds on Symbian OS, complementing it with a graphical user interface library and reference applications.

For licensees, Series 60 is a platform on which they can build their own feature-rich terminals. It takes as a base a large colour screen and an easy to use User Interface, and includes a rich suite of applications.

- Multimedia messaging
- Content Downloading
- Mobile browsing
- Native Symbian applications

Symbian components include data management, communications, graphics, multimedia, security, application engines, messaging engine, Bluetooth, browser engines and support for data synchronisation and internationalisation.

In May 2002, Nokia and Siemens announced an agreement – part of which included the licensing of Series 60 Platform by Siemens.

Future Nokia products based on Symbian OS
Nokia will launch new – Symbian OS based – products in the following categories:

- Communicator – building on the success of the Nokia 9000 Communicator, Nokia 9110 Communicator and Nokia 9200 Communicator series of phones
- Imaging phones – the first of which is the Nokia 7650
- Media phones – designed for consuming media and browsing
- Entertainment focussed phones

On 21.5.2001, Jorma Ollila, CEO Nokia said "Nokia expects 50% of its 3G phones to use the Symbian Operating System by 2004." – and these product plans are building towards this.

Commercial benefits for both operators and developers

The widespread establishment of Symbian OS will bring significant commercial benefits, both direct and indirect.

Operators

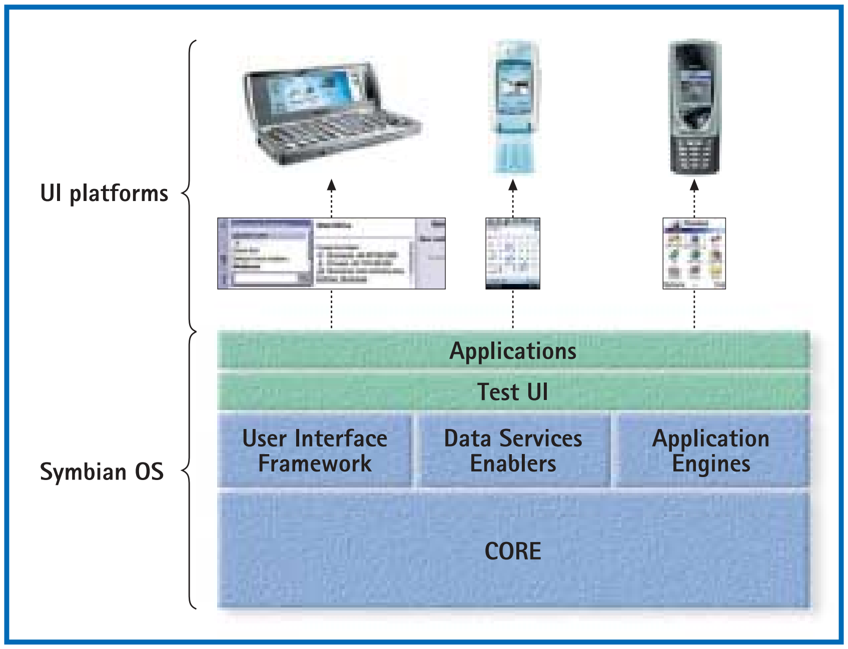
- Operators will benefit from having a wide pool of interoperable devices, built on open standards. They will be able to select from a wide range of terminal and infrastructure manufacturers with a rich set of interoperable solutions.
- In terms of value that operators can add, applications and content can all be more cost effectively supplied – given the common OS shared across phones.

Developers

- Developers will benefit from being able to target a greater number of consumers across one platform. Their porting and development costs will dramatically decline as the common OS means that applications will need to be developed only once.
- Applications can be written by virtually anybody. This software could be stand-alone, used only by the user of the device. However, just as easily, the software could be a networking application, enabling users to communicate with other users, or to access a resource somewhere in the internet.
- Equally, whilst costs are reduced, potential returns are increased as a wider pool of users is accessible – a win-win situation for all concerned.

Indirect benefits for the whole industry

- The above benefits assume that the number of users stays constant. In establishing Symbian OS, Nokia and the other industry players believe that there will be a Metcalfe effect – whereby the value of a network is the square of the number of users. As users proliferate, they will interact more, attracting even more users and consequently, more application developers, and content. This will benefit the whole industry.
- Symbian OS is the key to creation of this virtuous circle.



Symbian architectural overview (Table provided by Symbian)

Symbian OS – fundamental customer requirements

- It must work on stand alone portable devices
- It must work on different sorts of devices
- It must be future proof
- It must be open to all to license on fair and equal terms
- It must be open to all to develop applications – again with a level playing field for all
- It must be based on open standards

Perhaps the most important requirement is to work on a stand alone device. Symbian OS is fundamentally designed for mobile phones – with highly advanced features – but they must still function primarily as mobile phones.

This means that expectations are already set – for a user to consider buying Symbian OS based phones they must outperform a users current model in some areas and be at least equal in all others. The performance benchmark for Symbian OS is not the PC or portable computing devices but the phones that around one billion people already have in their pockets!

Symbian OS – architecture

Symbian OS architecture is designed to meet a number of requirements. It must be hardware independent so it can be used on a variety of phone types, it must be extendable so it can cope with future developments, and it must be open to all to develop for.

Architectural overview

- Core – Symbian OS core is common to all devices, i.e. kernel, file server, memory management and device drivers. Above this core, components can be added or removed depending on the product requirements.
- System Layer – the system layer provides communication and computing services such as TCP/IP, IMAP4, SMS and database management
- Application engines – above the Systems Layer sit the Application engines, enabling software developers (be they either employed by the phone manufacturer or independent) to create user interfaces to data.
- User Interface Software – can be made or licensed by manufacturers (for example in the case of the Nokia Series 60 platform).
- Applications – are slotted in above the User Interface.

Client Server Architecture

The power of the client-server framework is widely acknowledged in the software community. In Symbian OS, clients are programs that have user interfaces, and servers are programs that can only be accessed via a well-defined interface from other programs. The role of a client is to serve the user, while servers ensure timely response to all the clients while controlling the access to the resources of the actual system. Additionally, in practice, one server will often have many extra servers relying on the one original server.

Event management

Event management has long been considered a core strength of Symbian OS – reflecting the fact that Symbian OS was designed from the start to have event-based time sharing in a single thread. Rather than more conventional methods of having multi-threaded applications, Symbian OS enables the developer to think in terms of interactions and behaviours as the main artefacts. Enabling this shift from procedural to interactive designs has been one of the main challenges of modern software engineering – and this is one reason why Symbian OS has earned its reputation for advanced design.

Object oriented design

Because Symbian has an object-oriented design, it is easy to configure for different sorts of hardware, and being component-based, it allows manufacturers to add or remove components. This is crucial in enabling manufacturers to make devices that best suit their customers needs. This flexibility extends even to the user interface – again allowing a variety of different device designs to work from the same operating system.

For Symbian itself, the design allows new technology to be slotted into an already stable platform. This will provide a stable base as the telecommunications industry moves from 2G to 2.5G to 3G, with the further introduction of new technologies such as SyncML, Bluetooth, Multimedia Messaging amongst many. The picture will grow ever more complicated, especially when technologies are used in combination, but Symbian OS is ready!

For application developers, this separation of components allows them to program far richer applications – getting into the middle of the operating system.

Power management

Symbian OS users are used to the performance of mobile phones – and so demand similar performance in terms of weight and operating times when they adopt new devices.

Power management is built into the kernel of Symbian OS and is designed to make efficient use of the processor and peripherals and so minimise power usage. When peripherals are not being used they are switched off by the system. This lowers battery consumption, prolonging usage and allows for smaller batteries.

This meets the requirement to work on stand-alone portable devices – enabling manufacturers to make phones that capture the optimum combination of size and weight for their target market.

Robust and dependable

Symbian OS users will have experienced the performance levels achieved in this area by mobile phones. Devices should not lose user data, crash or require rebooting.

Symbian achieves this in two ways:
1. Each process runs in a protected address space – thus it is not possible for one application to overwrite another's address space.
2. The kernel also runs in a protected address space – so that a bug in one application cannot overwrite the kernel's stack or heap.

The client server architecture of Symbian OS allows applications to exchange data without compromising overall system integrity. This meets the requirement to work on stand alone portable devices – even though Symbian devices offer greatly enhanced functionality over standard mobile phones.

Memory management

For stand-alone portable devices, memory management is important. The need to minimise weight, device size and cost means the amount of memory available on a Symbian OS device is often quite limited.

Symbian OS always assumes that the memory available is limited, and minimises consumption at every turn. Consequently, less memory is actually required by the system. Also, having less memory helps to keep down power consumption.

Full multitasking

Symbian OS runs each application as a separate process – allowing multiple applications to run concurrently.
For instance, if a user is checking the calendar, and receives a call, the system must allow the user to switch between applications instantaneously. Equally, should the phone call result in an appointment, the user must be able to check the calendar – and still maintain the phone call. As phones become more data enabled, this ability will become ever more important.

An open operating system

Symbian OS is "open" – what does this mean?

Open to anyone to license
All manufacturers are treated equally – licensing Symbian OS is open to all on fair and equal terms.

Open to anyone to develop applications
The even-handed approach adopted towards manufacturers extends to developers. API's are made available as a matter of course. Support for 3rd party developers is a key tenet of Symbian OS so full SDKs and support are available for all products. Anyone can build an application for Symbian OS and again there is fair and equal access for all.

Based on open standards
Symbian focuses on one clear part of the value chain – providing a platform for all to build upon. Consequently Symbian avoids proprietary standards. It is an active participant in many standards forums – often drawing on the expertise of its shareholders and licensees.
The components of Symbian OS are based on agreed open standards.

Owned by the industry
Symbian has steadily increased the number of shareholders since it was inaugurated. With the addition of Siemens as the latest shareholder, Symbian shareholders now make over 70% of the phones sold globally. This breadth of ownership ensures that Symbian acts in the interests of the whole industry, driving open standards and promoting interoperability.

Symbian product releases – launching new technology

Symbian regularly releases updated versions of its software as new components are incorporated
(for instance Symbian OS v6.1 includes GPRS components). These updates are mostly developed in-house by Symbian but in some cases they license software from other sources including the licensee companies.

Symbian has trusted relationships with its shareholders and licensees – who are the leading players in the mobile industry – so new technology is extensively tested with licensees before it is launched.

Each product release is accompanied by Software Developer Kits for both C++ and Java developers, with full emulators for PC and cross compilers for installing applications on the device.

Writing applications for Symbian OS

C++
Symbian OS is written in C++, so it is natural fit to develop applications also in C++. This provides the developer with the most flexibility and scope. However, this flexibility brings with it complexity, and in some cases it may be more appropriate to develop an application in Java, which is also well supported on Symbian OS devices.

Symbian's use of C++ is efficient and thoroughly object-oriented. The design of the OS focuses on getting the most out of the limited hardware resources of mobile devices and this affects the way that code is written throughout the system including at the application level. This requires developers to get used to a few programming idioms that aren't common in other systems. However, these idioms help in making efficient use of the hardware resources, especially the very limited amount of memory.
They also help simplify some of the more difficult tasks in application development.

Some of the idioms are:
- the cleanup stack – a straightforward method for claiming back memory if a memory allocation fails part-way through a function.
- the rule that a C++ constructor cannot leave (i.e. cause an exception). This results in a two-phase construction system for objects (i.e. make a empty new object first, then allocate the memory in a second step) which makes the cleanup stack system keep working even for complicated class constructions.
- Various naming conventions. E.g. C, T and R type classes, L (leaving) and non-L functions. The conventions quickly tell the developer useful information about the class or method without having to look up the definition.

Multitasking

One of the major design decisions taken in developing Symbian OS was to optimise the system for efficient event handling from the ground up. Native Symbian OS programs are written from the viewpoint of the events that occur rather than the traditional programming model of a main control program that regularly polls for events and then performs the appropriate actions. This traditional model often requires multiple threads to be used to perform these actions and this results in the complicated problem of synchronising access to application resources.

Symbian OS multitasking system eliminates this problem by having only a single thread that responds to events as they happen. An Active Scheduler implements non-preemptive multi-tasking within the context of this single thread. The Active Scheduler catches events as they occur and then runs the appropriate Active Object for that event. The Active Object does the processing for that event and then returns control to the Active Scheduler. If several events occur in quick succession, they are stored and each Active Object is run in turn. There is a priority system to determine which Active Object should be run first, but if there is an Active Object already running it will run to completion before the next one can be run, even if the next one is of a higher priority. Thus we have multitasking that is non-preemptive. Since a Active Object function can't be preempted there is no need to use mutexes, semaphores, critical sections or any kind of synchronization to protect against the activities of other active objects in the thread. However, to keep the system responsive, the processing of each event must be quick so that control is returned in order for the next event to be processed.

Traditional multi-threading is also implemented in Symbian OS. Multiple applications and servers can be run simultaneously. Threads implement preemptive multi-tasking, so one thread can preempt another if it has to handle an event – for instance, the window server can handle a key-press event while an application is running, by preempting the running application thread. The ability of one thread to preempt another depends on thread priority. The most critical threads in the system are given the highest priorities – with the kernel, including device drivers, the highest priority of all.

Application Architecture

Symbian OS has an application architecture that helps developers manage the complexity of graphical user interface based applications. A Symbian OS application is made up several parts. An Application Engine that contains all the non-UI parts of an application, an Application UI that handles the application events coming from the user and calls the Engine, and then there is the Application View (or several views) which contains the actual windows and controls (e.g. buttons, text and graphics) that show on the screen of a Symbian OS device. The Application Architecture has a built-in Active Scheduler so that developers don't need to understand the ins and outs of the Active Object system when writing normal applications.

The tools that come with Symbian OS SDK can be used to generate an application with this basic structure. This provides the developer with a good guide for how to continue the development of the application.

Java

All Symbian OS devices have Java available on them. The higher end devices tend to have Personal Java and the more popular devices have MIDP Java.

Programming in Java for Symbian OS

Programming in Java for Symbian OS is the same as programming for Java on any other OS. Some phone manufacturers may implement additional device-specific APIs which are not part of the Java specification. However, all implementations will at least have all the required APIs for the particular Java profile that is being supported.

Personal Java + JavaPhone

PersonalJava (or pJava) is in many ways similar to Standard Java as embodied in JDK 1.1. The main difference is that certain functional components – packages, classes or methods – which are obligatory in Standard Java, are optional, so reducing the minimum ROM budget for handheld devices like PDAs and mobile phones. It also adds some functionality like timers. However Symbian has opted for a fairly complete PersonalJava API set, implementing virtually all optional components including RMI and JDBC support.

Over and above the standard Java functionality, Symbian OS also provides an implementation of the JavaPhone 1.0 API that gives Java programs access to a selection of invaluable native services on the phone. For example, with the Java Telephony API an application can create and terminate calls, listen for and answer incoming calls, detect changes in call state. There is also a Calendar and address book API, a P2P wireless datagram API for exchanging exchange datagrams with other devices via UDP or SMS, and javax.comm which allows Java programs to use the serial and infra-red ports on the phone.

MIDP Java
The Mobile Information Device Profile (MIDP) is a set of Java 2 Micro Edition (J2ME) APIs targeted at mobile information devices, such as mobile phones and two-way pagers. The MIDP specification addresses issues such as user interface, persistent storage, networking, and application model. It is generally implemented on the Connected Limited Device Configuration (CLDC) and it provides a basic J2ME application runtime environment.

MIDP applications (or midlets) run on the KJava Virtual Machine (KVM). This is a small-footprint, highly optimized virtual machine that meets the CLDC's minimum requirements. KVM implementations provide only a subset of the standard Java APIs. Limitations on memory management require that care is taken when writing applications and make it difficult to port or run larger applications.

Porting
Porting is feasible from a variety of sources:

Porting between Symbian OS devices
There are both different Symbian OS devices made by the same manufacturer, and devices made by different manufacturers. It is in the financial interests of developers to make their applications available for all devices and so maximise their potential market. An important feature of the design of Symbian OS is that the Core of the system is the same for all the devices that are based on the same OS release. The differentiation is in the Graphical User Interface style – the screen size varies between devices and there are also different input methods. This means the representation of the application on screen for one device may be very different than for another device. However, if an application is developed according to the application architecture mentioned above, then porting an application means only the GUI part of the code needs to be modified. The Application Engine can stay the same for all devices. The Application UI will probably stay the same if different device types were considered when it was developed; i.e. different input methods are allowed for. The Application View, which is the part the user actually sees will be the only part that requires significant change or rewriting. Within the same UI – for instance Series 60 platform, these changes are further reduced.

Porting from Palm or WinCe devices
Both Nokia and Symbian have documents covering this – contact Symbian or www.forum.nokia.com for more details.

Porting C applications
To aid in porting engine and communications code from other systems, the C Standard Library provides a Posix-compliant API set, layered on top of the more fundamental C++ APIs. However, it is best to re-write user interfaces in the standard Symbian OS way since this will make most efficient use of the resources of the device.

Development environment (SDK's)
Nokia provides SDK's for Nokia's Symbian-based phones to enable independent developers to create software for those phones. The Nokia 9200 Communicator Series SDK for Symbian OS is recommended when creating Nokia 9200 Communicator series optimized applications. The Nokia 9200 Communicator Series SDK for Symbian OS guides the implementation of Communicator specific sequences and commands, which in turn improves the end user experience. Nokia Series 60 SDK for Symbian OS is compatible with the Nokia 7650. Nokia Series 60 SDK for Symbian OS offers a great range of Application Programming Interfaces (APIs) to build software on. Part of the SDK includes documentation on these API's – not only describing the APIs but also offering examples of how to use them.

The main parts of an SDK are a device emulator that runs on the PC, a cross compiler for compiling software for the device and assorted tools that are required for application development.

There is also a large amount of documentation and plenty of example applications in the SDK that help a developer get started with using the system.

Symbian also provides support for 3rd party developers.

For more information contact www.forum.nokia.com or www.symbian.com.

Forum Nokia
Forum Nokia's Symbian-section and Business Opportunities- section provides developers with valuable information about development and business opportunities for Nokia devices running on Symbian OS. It is possible to download Software Development Kits, view documents to familiarise oneself with Nokia 9200 Communicator Series, Nokia 7650 and Series 60 Platform and also get information about Nokia sales channels.

For more information visit www.forum.nokia.com

Symbian and the Symbian Competence Centres have training courses for developers with different levels of experience. Developers new to Symbian OS can read one of the Symbian programming books that are available and try a few examples before going on the beginner course.

Symbian also has a developer focussed website www.symbian.com/developer where they host online discussions for Symbian OS developers and have a very useful knowledge base with answers for many problems that Java and C++ developers will come across.

Nokia Mobile Phones
P.O. Box 100
FIN-00045 NOKIA GROUP, Finland
Phone: +358 (0) 7180 08000
www.nokia.com